DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE CODE: 21CS2214AA

COURSE TITLE: DESIGN AND ANALYSIS OF ALGORITHMS

Lab 1:

Date of the Session: ___/___/___ **Time of the Session: _____to_____**

Pre-Lab Task:

1. Calculate the time complexity of the following:

```
int binarySearch(int a, int low, int high, int tar)
{
int mid;
if (low > high)
return (0);
mid = floor((low + high)/2)
if (a[mid] == tar)
        return (mid)
else
        if tar < a[mid]
                return binarySearch(a, low, mid-1, tar)
        else
                return binarySearch(a, mid+1, high, tar)
}
```

After 1st iteration : length of Array = $n/2$
After 2nd iteration : length of Array = $n/(2)^2$
After 3rd iteration : length of Array = $n/(2)^3$
After K th iteration : length of Array = $n/(2)^k$
At Kth iteration array size is 1. → $n/2^k = 1$ → $2^k = n$
Applying $\log_2$ on B.S. → $k\log_2 2 = \log_2 n$ → $k = \log_2 n$

2.

```
function fn(n)
{
if(n<0) return 0;
if(n<2) return n;
return fn(n-1)+fn(n-2);
}
```

	n=0	n<2	n>2			
0	0	0	0	0	0	0
1	0	0	0	0	0	0
1+1	1+0	1	1	2	1	1
0+1	0	1+1	1	0	2	1
1	0	0	1+1+1	0	0	1+1+1
				2	3	1+1+1+2

n = 0. → 2
n > 0 && n < 2 → t = 3
n > 2 then t = (1+1+1)+2
1 = then (n-1)
1 = then (n-2)

3) The median of a list of numbers is essentially its middle element after sorting. The same number of elements occur after it as before. Given a list of numbers with an odd number of elements, find the median?

Example

arr=[5,3,1,2,4]

The sorted array a'=[1,2,3,4,5]. The middle element and the median is 3.

Function Description

Complete the *findMedian* function in the editor below.

findMedian has the following parameter(s):

- *int arr[n]:* an unsorted array of integers

Returns

- *int:* the median of the array

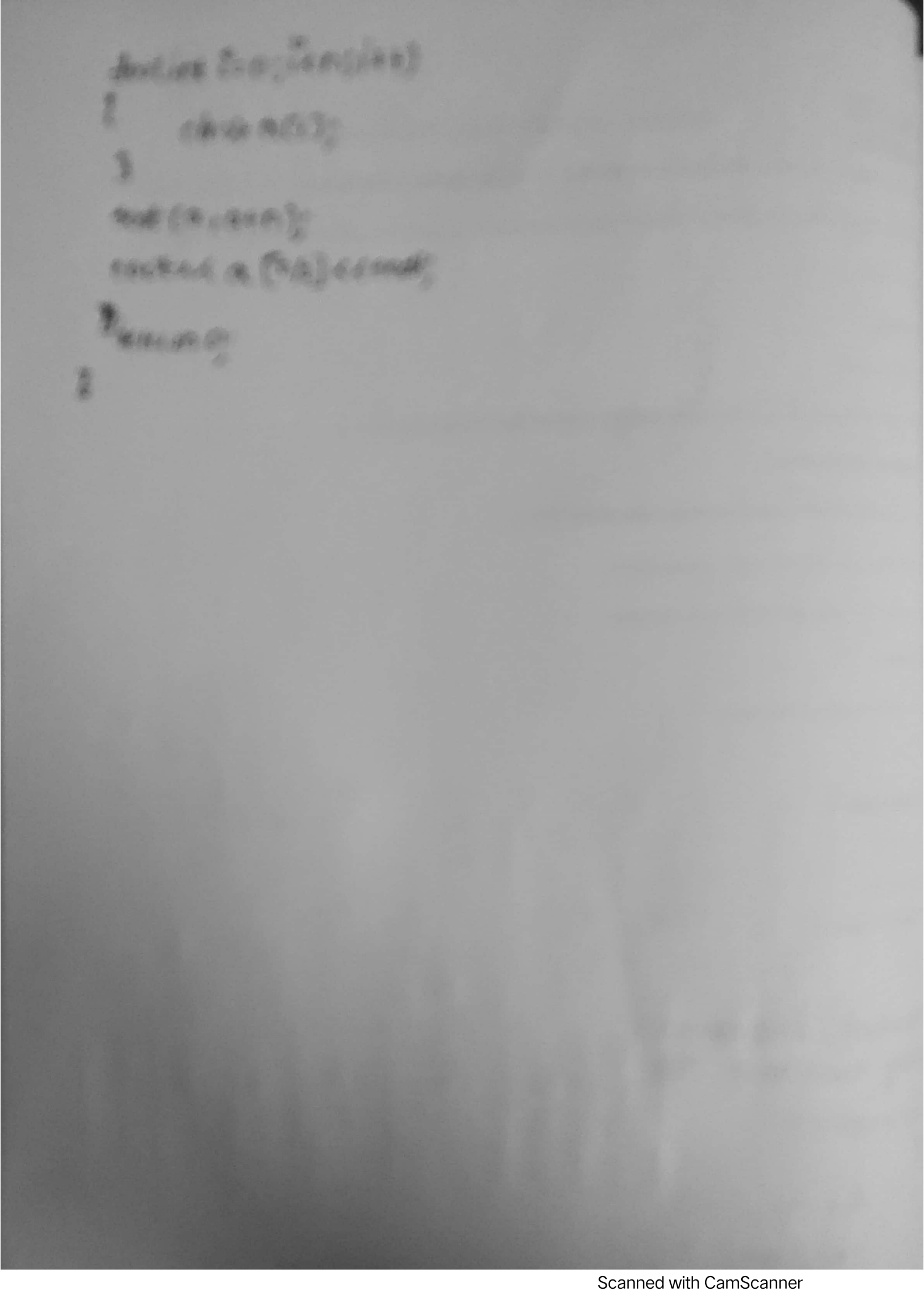
Sample Input 0

```
7
0 1 2 4 6 5 3
```

Sample Output 0

```
3
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int a[n];
```



```
for(int i=0;i<n;i++)
{
    cin>>a[i];
}
sort(a,a+n);
cout<<a[n/2]<<endl;
}return 0;
}
```

In-Lab Task:

1) Given an array of strings **arr[]**, the task is to sort the array of strings according to frequency of each string, in ascending order. If two elements have same frequency, then they are to be sorted in alphabetical order.

Input: arr[] = {"Ramesh", "Mahesh", "Mahesh", "Ramesh"}

Output: ("Mahesh","Ramesh"}

Explanation:
As both the strings have same frequency, the array is sorted in the alphabetical order.

```
#include<bits/stdc++.h>
using namespace std;

int comparator(pair<int, string> x, pair<int, string> y)
{
    if(x.first < y.first)
        return 1;
    else if (x.first > y.first)
        return 0;
    else
    {
        if(x.second < y.second)
            return 1;
        else
            return 0;
    }
}
void sortString (string str[], int n)
{
    unordered_map <string, int> m;
    for(int i=0; i<n; i++)
    {
        m[str[i]]++;
    }
```

```
    vector<pair<int, string>> vec;
    for(auto it = m.begin(); it != m.end(); it++)
    {
        vec.push_back(make_pair(it->second, it->first));
    }
    sort(vec.begin(), vec.end(), comparator);
    for(int i=0; i<vec.size(); i++)
    {
        cout<<vec[i].second<<", ";
    }
}
int main()
{
    int n;
    cin>>n;
    string str[n];
    for(int i=0; i<n; i++)
    {
        cin>>str[i];
    }
    sortString(str, n);
    return 0;
}
```

2) Given an array of strings **words[]** and the **sequential order** of alphabets, our task is to sort the array according to the order given. Assume that the dictionary and the words only contain lowercase alphabets.

Input: words = {"word", "world", "row"}, order = "worldabcefghijkmnpqstuvxyz"

Output: "world","word","row"

Explanation:
According to the given order 'l' occurs before 'd' hence the words "world" will be kept first.

```
#include <bits/stdc++.h>
using namespace std;
unordered_map<char, int> mp;
bool comparator(string &a, string &b)
{
    for(int i=0; i<min(a.size(),b.size()); i++)
    {
        if(mp[a[i]] != mp[b[i]])
            return mp[a[i]] < mp[b[i]];
    }
    return (a.size() < b.size());
}
void printSorted(vector<string> words, string order)
{
    for(int i=0; i<order.size(); i++)
    {
        mp[order[i]] = i;
    }
    sort(words.begin(), words.end(), comparator);
    for (auto x : words)
        cout << x << " ";
}
```

```
int main()
{
    vector<int> n;
        cin>>n;
    vector<string> words;
    string str;
    for(int i=0;i<n;i++)
    {
        cin>>str;
    }   words.push_back(str);
    string order;
    cin>>order;
    printSorted(words,order);
    return 0;
}
```

Post-Lab Task:

1) Given an array arr[] of N strings, the task is to sort these strings according to the number of upper-case letters in them try to use zip function to get the format.

Input arr[] = {poiNtEr, aRRAy, cOde, foR}

Output: [('cOde', 1), ('foR', 1), ('poiNtEr', 2), ('aRRAy', 3)]

"aRRAy" R, R, A->3 Upper Case Letters

"poiNtEr" N, E->2 Upper Case Letters

"cOde" O->2 Upper Case Letters

"foR" R->3 Upper Case Letters

```
#include <bits/stdc++.h>
using namespace std;
int countUpper(string str)
{
    int count=0;
    for(int i=0; i<str.length(); i++)
    {
        if(str[i]>='A' && str[i]<='Z')
            count++;
    }
    return count;
}
void sortUpper(string arr[], int n)
{
    vector<pair<int, string>> vp;
    for(int i=0; i<n; i++)
    {
        vp.push_back(make_pair(countUpper(arr[i]), arr[i]));
    }
```

```
    sort(vp.begin(), vp.end());
    for(int i=0; i<vp.size(); i++)
    {
        cout<<"("<<vp[i].second<<", "<<vp[i].first<<")"<<"
    }
}
int main()
{
    int n;
    cin>>n;
    string arr[n];
    for(int i=0; i<n; i++)
    {
        cin>>arr[i];
    }
    sortUpper(arr, n);
    return 0;
}
```

2) In KLU streets we have lots of electrical poles.
Note: all poles are sorted with respective to their heights.

Professor Hari Vege given the H = height of one pole to Mothi then asked him to print the position of that pole, here we consider index as a position. Mothi is particularly good at algorithms, so he written an algorithm to find that position. But he is extremely poor at finding time complexity. Your task is to help your friend Mothi to analyze the time complexity of the given problem.

```
Int BinarySearch (int a, int low, int high, int tar)
{
    int mid;
    if (low > high) return 0;
    mid = floor((low + high)/2)
    if (a[mid] == tar)
            return mid;
    else if (tar < a[mid])

            return BinarySearch (a, low, mid-1, tar)

    else
            return BinarySearch (a, mid+1, high, tar)
}
```

After 1st iteration : length of array → $n/2$

After 2nd iteration: length of array → $n/(2)^2$

After 3rd iteration : length of array → $n/(2)^3$

At kth iteration array size is 1 → $n/2^k = 1$

$2^k = n$

Applying $\log_2$ on B.S, → $k \log_2^2 = \log_2^n$

→ $k = \log_2^n$

(For Evaluator's use only)

Lab 2:

Date of the Session: ___/___/___ Time of the Session: _____to_____

Pre-Lab:

1) Given a text txt[0..n-1] and a pattern pat[0..m-1], write a function search (char pat [], char txt[]) that prints all occurrences of pat[] in txt[] using naïve string algorithm?(assume that n > m)

Input

txt[] = "THIS IS A TEST TEXT"

pat[] = "TEST"

Output

Pattern found at index 10

Input

txt[] = "AABAACAADAABAABA"

pat[] = "AABA"

Output

Pattern found at index 0

Pattern found at index 9

```
#include <bits/stdc++.h>
using namespace std;

void search (string pat, string str)
{
    int m= strlen (pat);
    int n= strlen (str);
    for(int i=0; i<= n-m; i++)
    {
        int j;
        for(j=0; j<m; j++)
        {
            if (str[i+j]!= pat[j])
                break;
```

```
        if (j == m)
        {
            cout << "pattern found at index " << i << endl;
        }
    }
}
int main()
{
    string str, pat;
    cin >> str >> pat;
    search(pat, str);
    return 0;
}
```

2) Discuss the Rabin Karp algorithm for string matching and explain time complexity of the algorithm?

Algorithm: Rabin-Karp-matches (T, P, d, q)

1. n = t.length
2. m = p.length
3. h = d^{m-1} mode (q is prime number)
4. p = 0
5. to = 0
6. for i = 1 to m do
7. p = (dp + P[i]) mod q
8. to = (dto + T[i]) mod q
9. for s = 0 to n-m do
10. if p == ts then
11. if P[1 ---- m] = T[s+1 - --- s+m] then
12. print " pattern occurs with shift s
13. if s < n-m then
14. t_{s+1} = (d (ts - T[s+1] h) + T[s+m+1]) mod q.

Time complexity:

worst case - O ((n-m+1)+m)
Average case - O (n+m)
Best case - O(n+m)

3) Stefan is a guy who is suffering with OCD. He always like to align things in an order. He got a lot of strings for his birthday party as gifts. He like to sort the strings in a unique way. He wants his strings to be sorted based on the count of characters that are present in the string.

Input

aaabbc

cbbaaa

Output

aabbcc

aabbcc

If in case when there are two characters is same, then the lexicographically smaller one will be printed first

Input:

aabbccdd

aabcc

Output:

aabbccdd

baacc

```
#include<bits/stdc++.h>
using namespace std;
void sortArr(string str)
{
    int n = str.length();
    int h[26]={0};
    for(int i=0; i<n; i++)
    {
        h[str[i]-'a']++;
    }
```

```
    vector<pair<int,char>> vp;
    for(int i=0;i<n;i++)
    {
        vp.push_back (h[str[i]-'a'], str[i]);
    }
    sort (vp.begin(), vp.end());
    for(int i=0; i<vp.size();i++)
        cout<< vp[i].second;
}
int main()
{
    string s;
    cin>>s;
    sortArr(s);
    return 0;
}
```

In-Lab:

1) Naive method and KMP are two string comparison methods. Write a program for Naïve method and KMP to check whether a pattern is present in a string or not. Using clock function find execution time for both and compare the time complexities of both the programs (for larger inputs) and discuss which one is more efficient and why?

Sample program with function which calculate execution time:

```
#include<stdio.h>
#include<time.h>
void fun()
{
        //some statements here
}
int main()
{
        //calculate time taken by fun()
clock_t t;
        t=clock();
fun();
        t=clock()-t;
        double time_taken=((double)t)/CLOCK_PER_SEC; //in seconds
printf("fun() took %f seconds to execute \n",time_taken);
return 0;
}
```

Program for naive method

```
void func()
{
    char txt[1000], pat[1000];
    scanf("%[^\n]s", txt, pat);
    int m = strlen(pat);
    int n = strlen(txt);
```

```
for(int i=0 ; i<n-m;i++)
{
    int j;
    for(j=0 ; j<m; j++)
    {
        if (txt[i+j] != pat[j])
            break.
    }
    if ( j==m)
    printf(" pattern found at index %d \n", i);
}
}
```

2) Andrea is working in a photo studio where his boss has given him a task to arrange the photos of family members. He is French and he do not know English somehow, he managed to send the list of names to you (his friend). Help Andrea to sort the photos.
(Note: implement the odd even merge algorithm)

Input

5

Neil Katherine Harry Stefan Dennis

Output

Dennis Harry Katherine Neil Stefan

```
package P;
import java.util.Scanner;
class Sortstrings {
    class public static void main(String[] args)
{
    int i, j, n, c=1;
    String temp;
    Scanner sc=new Scanner(System.in);
    n=sc.nextInt();
    String names[]=new String[n];
    for(i=0;i<n;i++)
    {
        name[i]=sc.next();
        c++;
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(names[j-1].compareTo(names[j])>0)
            {
                temp=names[j-1];
                names[j-1]=names[j];
```

```
            names[j] = temp;
        }
    }
}

for(i=0; i<n; i++)
    System.out.print(names[i] + "  ");
    }
}
```

Post-Lab:

1) Given a pattern of length- 5 window, find the valid match in the given text by step-by-step process using Robin-Karp algorithm
Pattern: 2 1 9 3 6

Modulus: 21

Index: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Text: 9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

hash value (Text) = 92721

hash value (pat) = 21936

h(Text) != h(pat)

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h[txt] = 2721

h[pat] = 21936

h[txt] = h[pat]:

9 2 7 2 1 8 3 0 5 7 1 2 1 9 3 6 2 3 9 7

h(txt) = 721183

h(pat) = 21936

h(txt) != h(pat).

9 2 7 21 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h(txt) = 21830

h(pat) = 21936

h(txt) != h(pat)

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h(txt) = 18305

h(pat) = 21936

h(txt) != h(pat).

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h(txt) = 30571

h(pat) = 21936

h(txt) != h(pat)

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h(txt) = 12193

h(pat) = 21936

h(txt) != h(pat).

9 2 7 2 1 8 3 0 5 7 1 2 1 2 1 9 3 6 2 3 9 7

h(txt) = 21936

h(pat) = 21936

found index 13.

2) James is sharing his information with his friend secretly in a chat. But he thinks that message should not understandable to anyone only for him and his friend. So he sent the message in the following format.

Input

a1b2c3d4e

Output

abbdcfdhe

Explanation:

The digits are replaced as follows:

- s[1] -> shift('a',1) = 'b'
- s[3] -> shift('b',2) = 'd'
- s[5] -> shift('c',3) = 'f'
- s[7] -> shift('d',4) = 'h'

```
a = str(input())
m = list(a)
d = []
for i in range(1, len(m), 2):
    d.append(m[i-1])
    t = int(m[i]) + ord(m[i-1])
    d.append(chr(t))
d.append(m[len(m)-1])
for i in range(len(d)):
    print(d[i], end="")
```

Lab 3:

Date of the Session: ___/___/___ **Time of the Session:** _____to_____

Pre-Lab-Task:

1) Calculate the time complexity of the following:

```
void multiply(int A[][N], int B[][N], int C[][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            C[i][j] = 0;
            for (int k = 0; k < N; k++)
            {
                C[i][j] += A[i][k]*B[k][j];
            }
        }
    }
}
```

for(int i=0; i<N; i++) ——— (N+1) times

for(int j=0; j<N; j++) ——— N(N+1) times

C[i][j]=0 ——— N(N) times.

for(int k=0; k<N; k++) ——— N(N(N+1)) times

C[i][j]+=A[i][k]*B[k][j]; ——— N(N(N)) times

$= (N+1) + N(N+1) + N(N) + N(N(N+1)) + N(N(N))$

$= N+1+N^2+N+N^2+N^3+N^2+N^3$

$= 2N^3 + 3N^2 + N + 1$

Time complexity = $O(N^3)$.

2) Given two arrays of integers, find which elements in the second array are missing from the first array.

Example:

A={7,2,5,3,5,3}

B={7, 2, 5, 4, 6, 3, 5, 3}

The 'B' array is the original list. The numbers missing are {4, 6}.

Sample Input

10

203 204 205 206 207 208 203 204 205 206

13

203 204 204 205 206 207 205 208 203 206 205 206 204

Sample Output

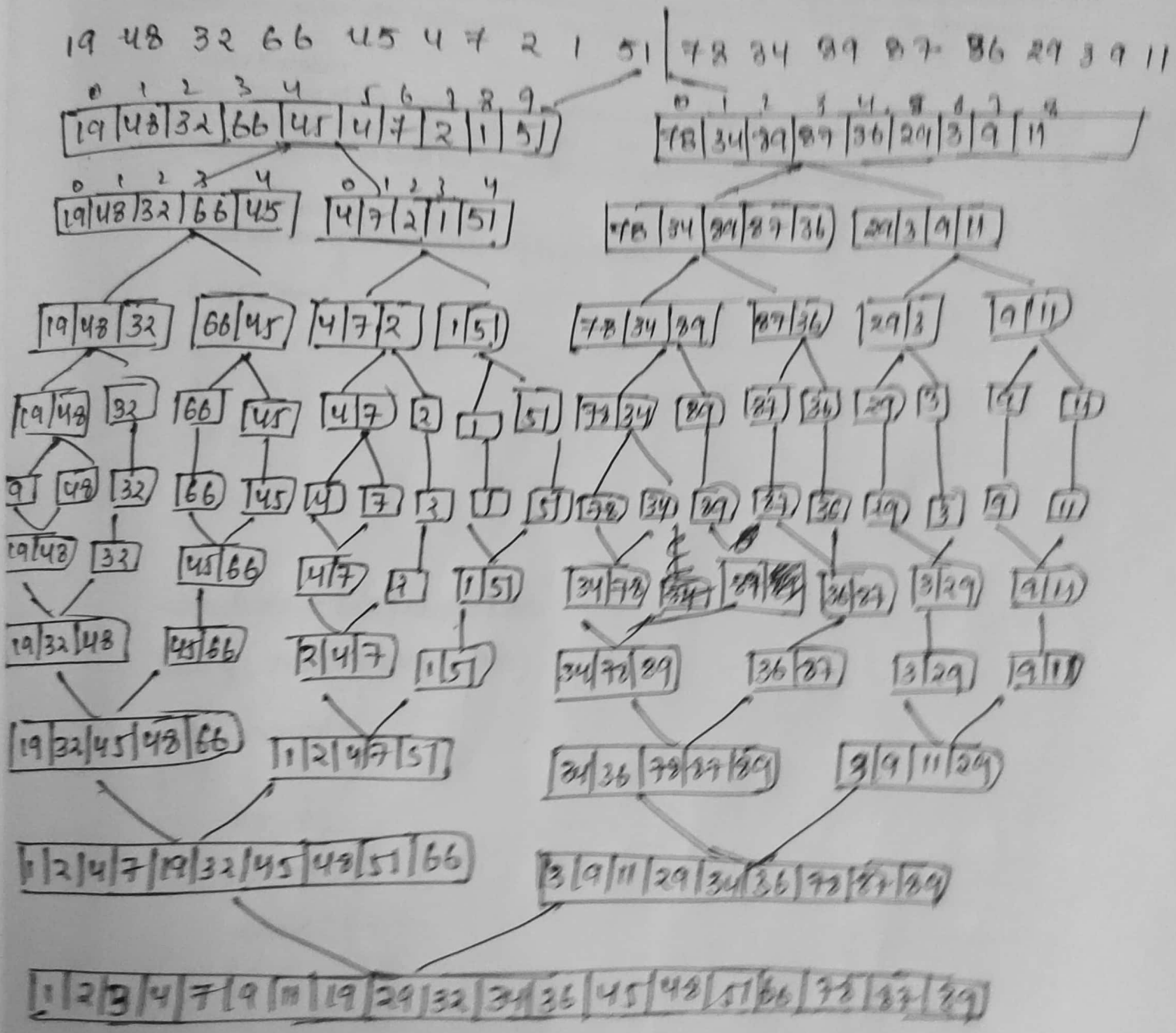
204 205 206

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
     int a[n];
    for(int i=0;i<n;i++)
        cin>>a[i];

    int m;
     cin>>m;
     int b[m];
    for(int i=0;i<m;i++)
       cin>>b[i];
```

```
int h1[1000]={0},h2[1000]={0};
for(int i=0;i<n;i++)
{
    h1[a[i]]++;
}
for(int i=0;i<m;i++)
{
    h2[b[i]]++;
}
sort(b,b+m);
for(int i=0;i<m;i++)
{
    if(h1[b[i]]<h2[b[i]])
    {
        cout<<b[i]<<" ";
        h1[b[i]]=h2[b[i]];
    }
}
}
```

3) Trace out the output of the following using Merge sort. 19, 48, 32, 66, 45, 4, 7, 2, 1, 51, 78, 34, 89, 87, 36, 29, 3, 9, 11.



In-Lab Task:

1) You are given an array A. You can decrement any element of the array by *1*. This operation can be repeated any number of times. A number is said to be missing if it is the smallest positive number which is a multiple of *2* that is not present in the array A. You must find the maximum missing number after all possible decrements of the elements.

Input Format:
The first line of input contains T denoting number if test cases.
The first line of each test case contains N, the size of the array.
The second line of each test case contains N space seperated intergers.

Output Format:
Print the answer for each test case in a new line.

Sample Input:
2
6
1 3 3 3 6 7
3
3 0 2

Sample Output:
8
4

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        long int a[n];
        for(int i=0;i<n;i++)
            cin>>a[i];

    sort(a, a+n);
```

```
int k=1;
for(int i=0;i<n;i++)
{
    if(a[i]>=2*k)
    {
        a[i] = 2*k;
        k++;
    }
}
if(a[n-1]%2)
{
    cout<<arr[n-1]+1<<"\n";
}
else
{
    cout<<arr[n-1]+2<<"\n";
}
return
}
return 0;
}
```

2) Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right.

Example

A= [5, 6, 8, 11]

8 is between two subarrays that sum to 11.

A=[1]

The answer is [1] since left and right sum to 0.

You will be given arrays of integers and must determine whether there is an element that meets the criterion. If there is, return YES. Otherwise, return NO.

Sample Input 0

```
2
3
1 2 3
4
1 2 3 3
```

Sample Output 0

```
NO
YES
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        int a[n];
        for(int i=0;i<n;i++)
        {
            cin>>a[i];
```

```
int x=0, y=0;
for(int i=0; i<n; i++)
{   cin>>a[i];
    y+=a[i];
}
y-=a[0];
if(n==1)
    cout<<"Yes\n";
else
{
    int i;
    for(i=0; i<n-1; i++)
    {
        if(x==y)
        {  cout<<"Yes\n";
           break;
        }
        x+=a[i];
        y-=a[i+1];
    }
    if(i==n-1)
        cout<<"No\n";
}
}
return 0;
}
```

Post-Lab Task:

1) Given a list of N array elements apply Merge sort.

*Note: Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.

Input Format

- The first line contains an integer, N, the number of elements in Array.
- The second line contains N space-separated integers.

Constraints

1<= N <= 1000

-1000 <= array elements <= 1000

Output Format

Print the array as a row of space-separated integers each iteration

Sample Input 0

```
10
10 1 9 2 8 3 4 7 5 6
```

Sample Output 0

```
1 10
1 9 10
2 8
1 2 8 9 10
3 4
3 4 7
5 6
3 4 5 6 7
1 2 3 4 5 6 7 8 9 10
```

```
#include<bits/stdc++.h>
using namespace std;
void merge(int collection[], int p, int q, int r, int series)
{
    int n1 = q-p+1;
    int n2 = r-q;
    int L[n1], M[n2];
    for(int i=0; i<n1; i++)
        L[i] = collection[p+i];
```

```
for (int j=0; j<n2; j++)
    m[j] = collection[q+p+j];
    int i, j, k;
    i=0, j=0, k=p;
    if (series==1)
    {
      while (i<n1 && j<n2)
      {
        if (k[i] <= m[j])
        {
          collections[k] = L[i]
          i++;
        }
        else
        {
          collection[k] = m[j];
          j++;
        }
        k++;
      }
    }
    else
    {
    while (i<n1 && j<n2)
    {
      if (L[i] >= m[j])
      {
        collection[k] = L[i];
        i++;
      }
      else
      {
        collection[k] = m[j];
        j++;
      }
    k++;
    }
    while (i<n1)
    {
      collection[k] = L[i];
      i++;
    k++;
    }
    while (j<n2)
    { collection[k] = m[j];
      j++;
      k++;
    }
    void mergesort (int arr[], int lb,
                    int ub, int seq)
    {
      if (lb < ub)
      {
        int mid = (lb+ub) >> 1;
        mergesort (arr, lb, mid, seq);
        mergesort (arr, mid-1, ub, seq);
        mergesort (arr, lb, mid, ub, seq);
      }
    }
    void printArray (int arr, int n)
    { for (int i=0; i<n; i++)
        cout << arr[i] << " ";
    }
    Import maine()
    {
      int n;
      cin >> n;
      int arr[n], i;
      for (i=0; i<n; i++)
        cin >> arr[i];
    int scale = size of (arr) /
                size of (arr[0]);
    mergesort (arr, 0, scale-1, n);
    printArray (arr, scale);
      return 0;
    }
```

2)

HackerLand National Bank has a simple policy for warning clients about possible fraudulent account activity. If the amount spent by a client on a particular day is *greater than or equal to2X* the client's median spending for a trailing number of days, they send the client a notification about potential fraud. The bank doesn't send the client any notifications until they have at least that trailing number of prior days' transaction data.

Given the number of trailing days d and a client's total daily expenditures for a period of n days, determine the number of times the client will receive a notification over all n days.

Example

expenditure=[10,20,30,40,50]

d=3

On the first three days, they just collect spending data. At day 4, trailing expenditures are [10,20,30]. The median is 20 and the day's expenditure is 40. Because 40>=2X20, there will be a notice. The next day, trailing expenditures are [20, 30, 40] and the expenditures are 50. This is less than 2X30 so no notice will be sent. Over the period, there was one notice sent.

Note: The median of a list of numbers can be found by first sorting the numbers ascending. If there is an odd number of values, the middle one is picked. If there is an even number of values, the median is then defined to be the average of the two middle values.

Sample Input 0

STDIN	Function
9 5	expenditure[] size n =9, d = 5
2 3 4 2 3 6 8 4 5	expenditure = [2, 3, 4, 2, 3, 6, 8, 4, 5]

Sample Output 0

2

```
from bisect import bisect_left, insort_left
n, d=map(int, input().split())
t=list(map(int, input().split()))
not i=0
last d= sorted(t[d]);
def med(d):
return last d[d//2] if d%2==1
else
((last(d//2) + last[d/2-1]))/2)
for i in range(d,n)
if t[i]>= 2*med()
not i+=1
def last d( bisect.left(last d, t[i-d]))
insort_left(last d, t[i])
print(not i)
```

Lab 4:

Date of the Session: 02/03/22 **Time of the Session:** 11:00 to 12:40 PM

Pre-Lab Task:

1) Trace the output of the following matrix multiplication using Strassen's Multiplication Method

A		B			C	
a	b	e	f	=	ae+bg	af+bh
c	d	g	h		ce+dg	cf+dh

A, B and C are the Matrices of Size NxN

a, b, c and d are the sub-Matrices of A of size N/2xN/2

e, f, g and h are the sub-Matrices of B of size N/2xN/2

A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

C

C_{11}	C_{12}
C_{21}	C_{22}

Strassen's matrix

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) B_{11}$

$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (c - d)(g + h)$.

Calculation

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$C_{11} = P + S - T + V$

$C_{22} = P + R - Q - U$

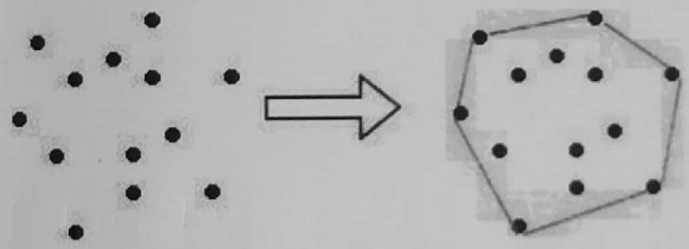
$C_{12} = R + T$

$C_{21} = Q + S$

2) Write a divide and conquer algorithm for finding the maximum and minimum in the sequence of numbers. Find the time complexity.

```
maximum (i, j, max, min)
{ if (i==j) {max=min=a[i];}
  elseif (i==j-1)
  { if(a[i]<a[j]) { max=a[j]; min=a[i];}
    else { max=a[i]; min=a[j];}}
    else { mid=[i+j] >>1; max, imum(i, mid, max, min);
maximum (mid+1; max1, min1);
if(max<max1) max=max1; else {min=min1;}}
```

3) Given an input is an array of points specified by their x and y co-ordinates. The output is the convex hull of this set of points by using Divide and Conquer algorithm.



Input : points[] = {(0, 0), (0, 4), (-4, 0), (5, 0),
(0, -6), (1, 0)};
Output : (-4, 0), (5, 0), (0, -6), (0, 4)

```
#include<bits/stdc++.h>
using namespace std;
pair<int,int> mid;
int quad(pair<int,int>p)
{
   if( p.first >= 0 && p.second>=0)
      return1;
   if( p.first <=0 && p.second >=0)
      return 2;
   if(p.first <=0 && p.second <=0)
      return 3;
   return4;
}
```

```
int orientation (pair<int,int> a, pair<int,int> b, pair<int,int>)
{ int res ((b.second - a.second) * (c.first - b.first)) -
        ((c.second - b.second) * (b.first - a.first)),
  if (res == 0)
     return 0;
  if (res > 0)
     return 1;
  return -1;
}
bool compare (pair<int,int> p, pair<int,int> a1)
{ pair<int,int> p = make_pair (p.first - mid.first, p.second -
                               mid.second)
  pair<int,int> a = make_pair (a1.first - mid.first, a1.second - mid.
                               second);
  int one = quad (p);
  int two = quad (a);
  if (one != two)
     return (one < two);
  return (p.second * a.first < a.second * p.first);
}
vector<pair<int,int>> merge (vector<pair<int,int>> a,
                             vector<pair<int,int>> b)
{ int n1 = a.size ();
  int n2 = b.size ();
  int ia = 0, ib = 0;
  for (int i = 1; i < n; i++)
    if (a[i].first > a[ia].first)
       ia = i;
  for (int i = 1; i < n2; i++)
    if (b[i].first > b[ib].first)
       ib = i;
  int inda = ia, indb = ib;
```